EMBEDDED
LINUX
CONFERENCE

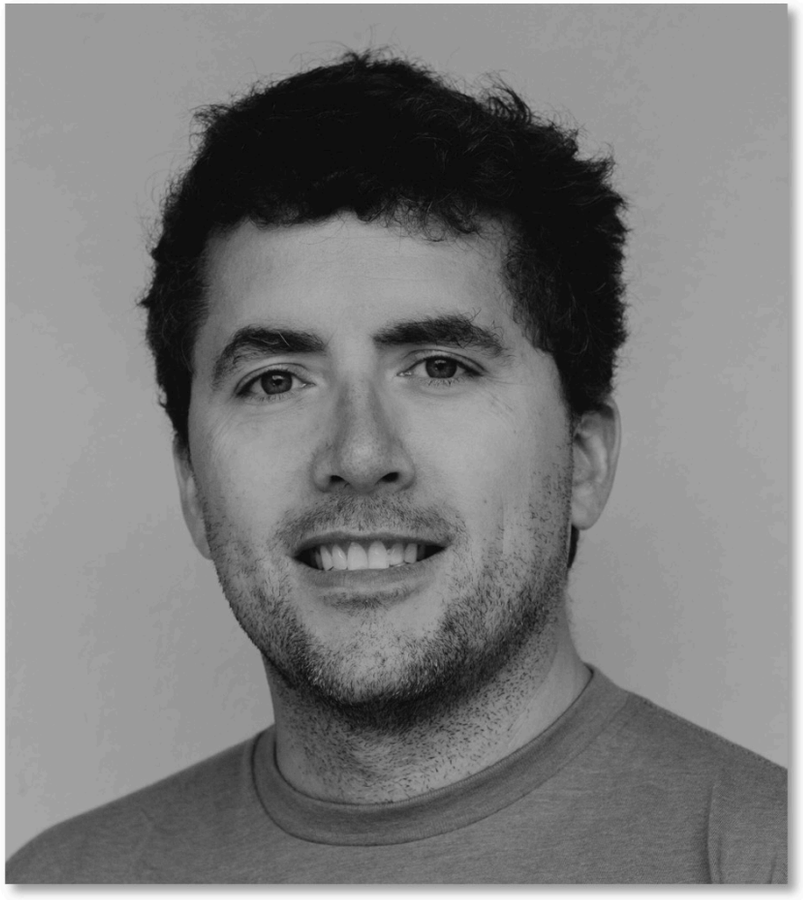
`unwrap()`-ing Rust on embedded Linux

Thomas Sarlandie

```rust
//! memfaultd
//! A Linux agent to capture system metrics, report crashes and
//! install updates over the air..

fn main() {
    try_adopt_rust().unwrap();
}
```

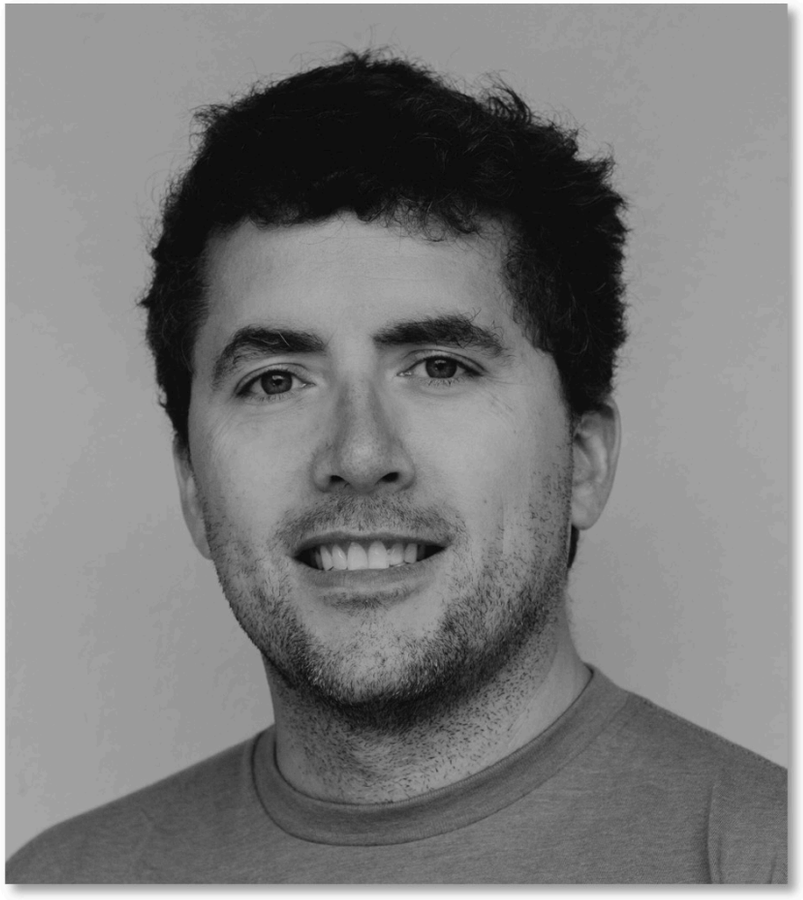
```rust
//! memfaultd
//! A Linux agent to capture system metrics, report crashes and
//! install updates over the air..

fn main() {
    try_adopt_rust().unwrap();
}
```

```rust
fn main() {
    try_adopt_rust().unwrap();
}

fn try_adopt_rust() ->
        Result<(), Box<dyn std::error::Error>> {
    // Can we compile Rust for all supported platforms?
    get_toolchain_and_compiler()?;

    // How long to learn Rust? What do we not know?
    write_code()?;

    // How big will it be? How fast will it run?
    optimize()?;
}
```



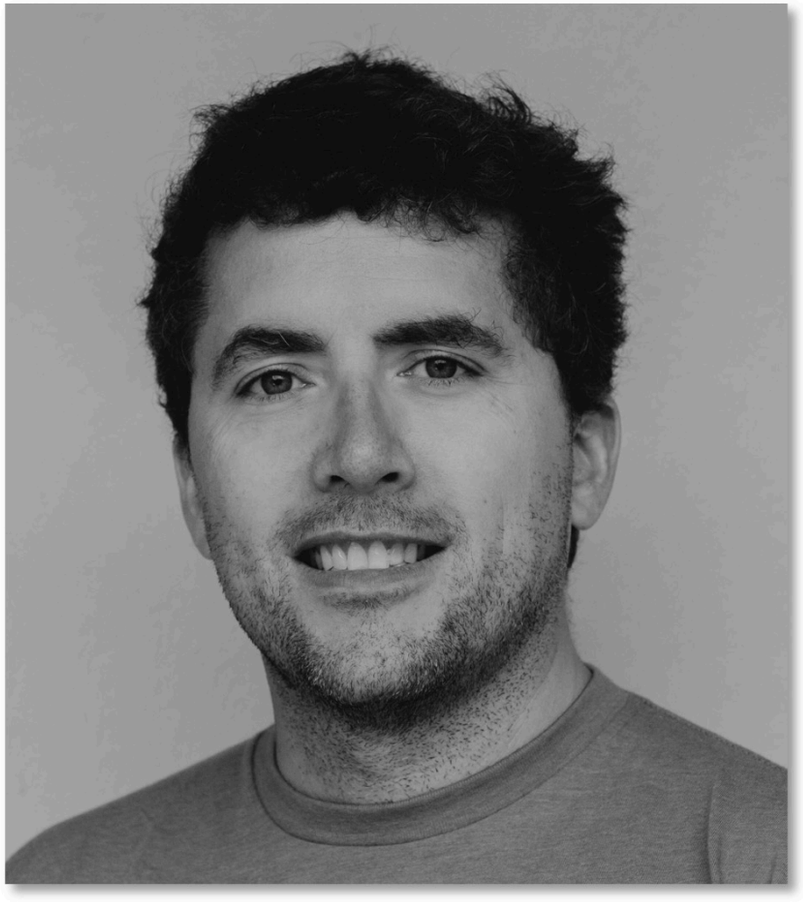
```
> rustup show

Your host: Thomas Sarlandie - @sarfata
Job title: Field CTO @Memfault

experience
----------

linux-appliances
dev-evangelism
sw-eng-management
Pebble - Fitbit

active job
----------
```

Memfault



```
 2
 3 Your host: Thomas Sarlandie - @sarfata
 4 Job title: Field CTO @Memfault
 5
 6 experience
 7 ----------
 8
 9 linux-appliances
10 dev-evangelism
11 sw-eng-management
12 Pebble - Fitbit
13
14 active job
15 ----------
16
```

Memfault



```
 3 Your host: Thomas Sarlandie - @sarfata
 4 Job title: Field CTO @Memfault
 5
 6 experience
 7 ----------
 8
 9 linux-appliances
10 dev-evangelism
11 sw-eng-management
12 Pebble - Fitbit
13
14 active job
15 ----------
16
17 help embedded companies ship better firmware
```

Memfault

get_tools()

Will it run?

Will Rust run on all the platforms I need to support?

Will Rust run on all the platforms I need to support?

```
$ rustc --print target-list |wc -l
237

$ rustc --print target-list |grep -i linux |wc -l
75
```

Will Rust run on all the platforms I need to support?

```
$ rustc --print target-list |wc -l
237

$ rustc --print target-list |grep -i linux |wc -l
75
```

Will Rust run on all the platforms I need to support?

```
$ rustc --print target-list |wc -l
237

$ rustc --print target-list |grep -i linux |wc -l
75
```

How do I cross-compile Rust code for my target?

```
$ cargo build --target=armv7-unknown-linux-gnueabihf
// 😕 Requires C cross-compilation environment
// 🚨 Difficult build environment to reproduce ...
```

How do I cross-compile Rust code for my target?

```
$ cargo build --target=armv7-unknown-linux-gnueabihf
// 😕 Requires C cross-compilation environment
// 🚨 Difficult build environment to reproduce ...
```

How do I cross-compile Rust code for my target?

```
$ cross build --target armv7-unknown-linux-gnueabihf
// Docker based
// Supports custom build environments with dynamic libraries
```

How do I cross-compile Rust code for my target?

```
$ cross build --target armv7-unknown-linux-gnueabihf
// Docker based
// Supports custom build environments with dynamic libraries
```


yocto
PROJECT

- meta-rust
  - Started by Cody Shafer in 2014
  - Merged by Randy MacLeod in Yocto 3.4 Honister
  - Provides a `cargo` class to build rust crates
  - Builds the rust compiler from source
  - Requires explicit dependency list (via `cargo-bitbake`)
  - `meta-rust/meta-rust` remains available for a more recent version

Yocto	Rust
Honister	1.54
Kirkstone	1.59
Langdale	1.63
Mickledore	1.68.2
Scarthgap	1.75

- meta-rust-bin
  - A project of the rust-embedded working group
  - Uses pre-compiled binaries for `rustc`, `cargo`, and `libstd-rs`
  - Compatible with all versions of Yocto since `dunfell`
  - Uses a different class name (`cargo_bin`) and can be used in parallel to `meta-rust`

meta-rust	meta-rust-bin
Yocto Official	Rust WG Official
Builds from source (Slower)	Uses pre-compiled binaries
Requires explicit dependencies	Automatically fetches dependencies
Locked version of Rust	Latest version of Rust

Shameless plug: interrupt.memfault.com/blog/rust-in-yocto

meta-rust	meta-rust-bin
Yocto Official	Rust WG Official
Builds from source (Slower)	Uses pre-compiled binaries
Requires explicit dependencies	Automatically fetches dependencies
Locked version of Rust	Latest version of Rust

Rust on Yocto? ✅

Shameless plug: interrupt.memfault.com/blog/rust-in-yocto

```
write_code()
```

Adoption Strategy

Adoption Strategy

rusting the C code - one file at a time

mylib.c

```
extern "C" void do_something();

void main_loop() {
    while (1) {
        do_something();
    }
}
```

mylib.c

```c
extern "C" void do_something();

void main_loop() {
    while (1) {
        do_something();
    }
}
```

main.rs

```rust
extern "C" fn main_loop();

fn main() {
    main_loop();
}

extern "C" fn do_something() {
    // Do something
}
```

build.rs

```
// Example custom build script.
fn main() {
  // Tell Cargo that if the given file changes, to rerun this build script.
  println!("cargo::rerun-if-changed=src/hello.c");
  // Use the `cc` crate to build a C file and statically link it.
  cc::Build::new()
    .file("src/hello.c")
    .compile("hello");
}
```

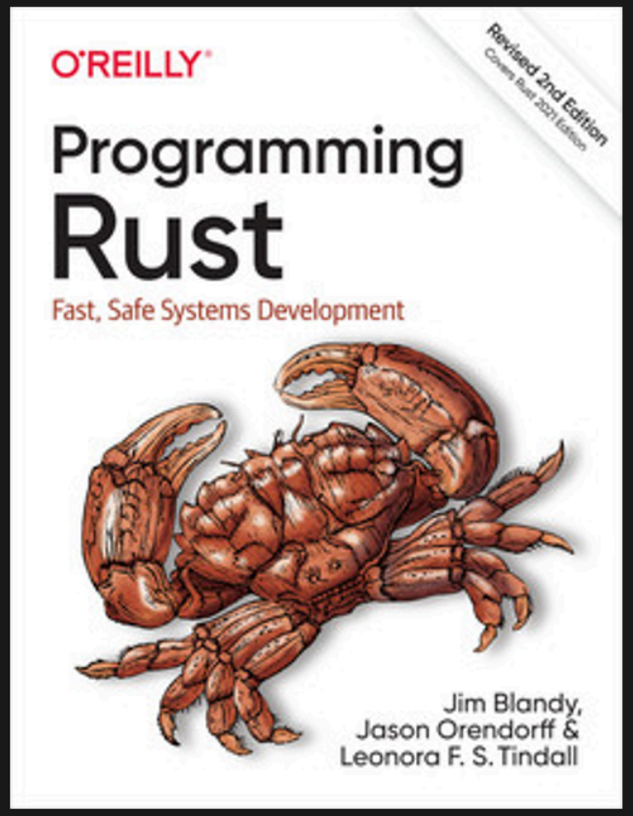
build.rs

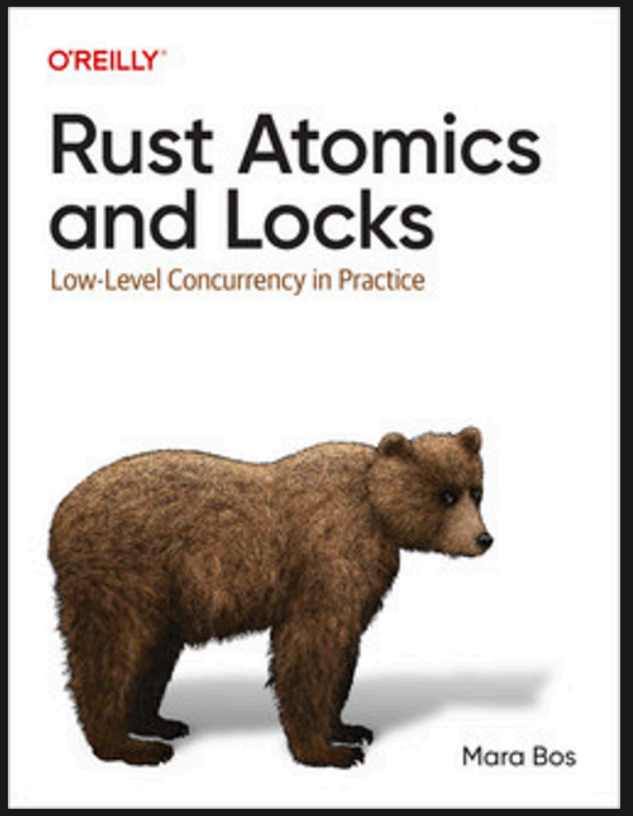
```rust
// Example custom build script.
fn main() {
  // Tell Cargo that if the given file changes, to rerun this build script.
  println!("cargo::rerun-if-changed=src/hello.c");
  // Use the `cc` crate to build a C file and statically link it.
  cc::Build::new()
    .file("src/hello.c")
    .compile("hello");
}
```

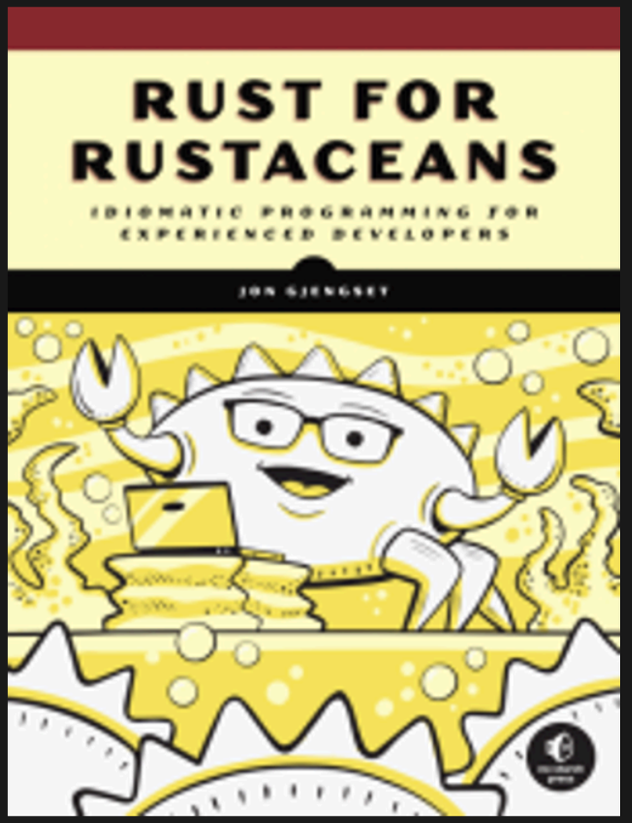
For more complicated libraries – see the `cmake` crate.

Learning Some Rust

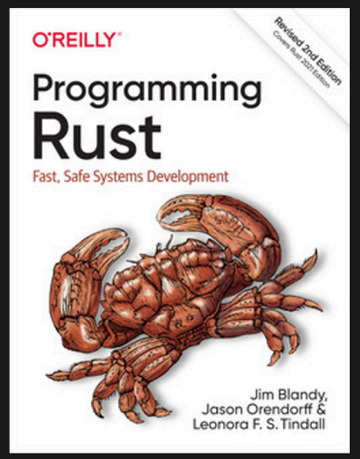
Learning Some Rust

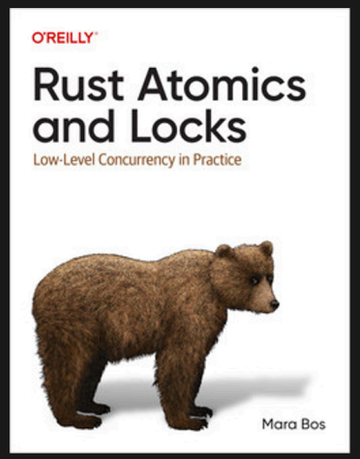






Learning Some Rust





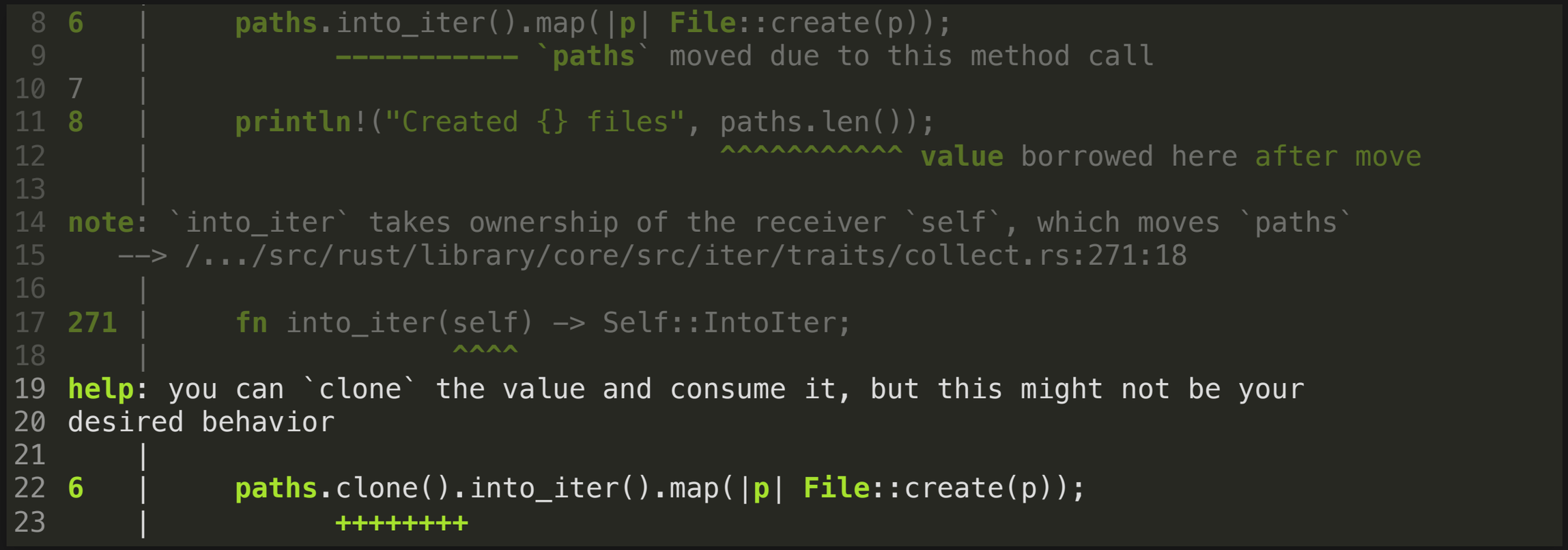
- The Rust Programming Language book
- doc.rust-lang.org/std => `Vec`, `String`, `PathBuf`, `Option`, `Result`, etc.
- Advent Of Code, YouTube
- Get Help - 🙏 James Munns

Lessons learnt ...

the Borrow Checker

```
error[E0382]: borrow of moved value: `paths`
   --> src/main.rs:8:34
    |
4   |     let paths: Vec<PathBuf> = vec![];
    |         ----- move occurs because `paths` has type `Vec<PathBuf>`,
    |               which does not implement the `Copy` trait
5   |
6   |     paths.into_iter().map(|p| File::create(p));
    |           ----------- `paths` moved due to this method call
7   |
8   |     println!("Created {} files", paths.len());
    |                                   ^^^^^^^^^^^ value borrowed here after move
    |
note: `into_iter` takes ownership of the receiver `self`, which moves `paths`
   --> /.../src/rust/library/core/src/iter/traits/collect.rs:271:18
    |
```

```
5   |
6   |     paths.into_iter().map(|p| File::create(p));
    |           ----------- `paths` moved due to this method call
7   |
8   |     println!("Created {} files", paths.len());
    |                                   ^^^^^^^^^^^ value borrowed here after move
    |
note: `into_iter` takes ownership of the receiver `self`, which moves `paths`
   --> /.../src/rust/library/core/src/iter/traits/collect.rs:271:18
    |
271 |     fn into_iter(self) -> Self::IntoIter;
    |                  ^^^^
help: you can `clone` the value and consume it, but this might not be your
desired behavior
    |
6   |     paths.clone().into_iter().map(|p| File::create(p));
```


6 | paths.into_iter().map(|p| File::create(p));
| ---------- `paths` moved due to this method call
7 |
8 | println!("Created {} files", paths.len());
| ^^^^^^^^^^^^ value borrowed here after move
|
note: `into_iter` takes ownership of the receiver `self`, which moves `paths`
--> /.../src/rust/library/core/src/iter/traits/collect.rs:271:18
|
271 | fn into_iter(self) -> Self::IntoIter;
| ^^^^
help: you can `clone` the value and consume it, but this might not be your
desired behavior
|
6 | paths.clone().into_iter().map(|p| File::create(p));
| ++++++++

```
error[E0382]: borrow of moved value: `paths`
   --> src/main.rs:8:34
    |
4   |     let paths: Vec<PathBuf> = vec![];
    |         ----- move occurs because `paths` has type `Vec<PathBuf>`,
    |               which does not implement the `Copy` trait
5   |
6   |     paths.into_iter().map(|p| File::create(p));
    |           ------------ `paths` moved due to this method call
7   |
8   |     println!("Created {} files", paths.len());
    |                                   ^^^^^^^^^^^ value borrowed here after move
    |
note: `into_iter` takes ownership of the receiver `self`, which moves `paths`
   --> /.../src/rust/library/core/src/iter/traits/collect.rs:271:18
    |
```

Take your time - Read the error message

beginner Rust

```rust
/// Memfault Network client
pub struct Client<'a> {
    client: blocking::Client,
    config: &'a MemfaultdConfig,
    device_info: &'a DeviceInfo,
}

impl<'a> Client<'a> {
    pub fn new(config: &'a MemfaultdConfig,
               device_info: &'a DeviceInfo)
            -> Result<Self> {
        // ...
    }
}
```

beginner Rust

```rust
/// Memfault Network client
pub struct Client<'a> {
    client: blocking::Client,
    config: &'a MemfaultdConfig,
    device_info: &'a DeviceInfo,
}

impl<'a> Client<'a> {
    pub fn new(config: &'a MemfaultdConfig,
               device_info: &'a DeviceInfo)
            -> Result<Self> {
        // ...
    }
}
```

🤓 Look Ma! I can use lifetimes!

No un-necessary copies!

pro Rust

```rust
/// Memfault Network client
pub struct NetworkClientImpl {
    client: blocking::Client,
    config: NetworkConfig,
}


impl NetworkClientImpl {
    pub fn new(config: NetworkConfig) -> Result<Self> {
        // ...
    }
}
```

pro Rust

```rust
/// Memfault Network client
pub struct NetworkClientImpl {
    client: blocking::Client,
    config: NetworkConfig,
}


impl NetworkClientImpl {
    pub fn new(config: NetworkConfig) -> Result<Self> {
        // ...
    }
}
```

 Avoid early optimization - Value simplicity

Rust will protect you from **unexpected behavior**

Rust will protect you from **unexpected behavior**

(as long as you stay away from `unsafe()`).

But your program can still crash ...

```
serde_json::from_str::<Value>("{ key: 42 }").unwrap();
```

But your program can still crash ...

```
    serde_json::from_str::<Value>("{ key: 42 }").unwrap();
```

```
$ cargo run
thread 'main' panicked at 'called `Result::unwrap()` on an `Err` value:
    Error("key must be a string", line: 1, column: 3)'
```

But your program can still crash ...

```
    serde_json::from_str::<Value>("{ key: 42 }").unwrap();
```

```
$ cargo run
thread 'main' panicked at 'called `Result::unwrap()` on an `Err` value:
    Error("key must be a string", line: 1, column: 3)'
```

```
    let v = vec![0];
    v[42]
```

But your program can still crash ...

```
    serde_json::from_str::<Value>("{ key: 42 }").unwrap();
```

```
$ cargo run
thread 'main' panicked at 'called `Result::unwrap()` on an `Err` value:
    Error("key must be a string", line: 1, column: 3)'
```

```
    let v = vec![0];
    v[42]
```

```
$ cargo run
thread 'main' panicked at 'index out of bounds:
    the len is 1 but the index is 42'
```

- Treat `unwrap()`, `expect()`, etc as you would asserts in C.
- Understand what happens when a *Thread* panics.
- Have a strategy for dealing with crashed threads.

Concurrency

- Embrace Memory Safe concurrency!

- Choose one paradigm and stick to it
  - Shared memory with `Arc<Mutex<T>>`
  - Worker pool: Many threads working on one list of tasks
  - Actors: Independent threads communicating via messages

- The borrow checker does not protect you from deadlocks

optimize()

don't panic!

```
$ cargo init
$ cat src/main.rs
fn main() {
    println!("Hello, world!");
}
$ cargo build --release
$ du -sh target/release/helloworld
4.4M    target/release/helloworld
```

don't panic!

```
$ cargo init
$ cat src/main.rs
fn main() {
    println!("Hello, world!");
}
$ cargo build --release
$ du -sh target/release/helloworld
4.4M    target/release/helloworld
```

```
$ cat > Cargo.toml <<EOF
[profile.release]
# Strip your binaries of debug information
strip = true

# Optimize for size
opt-level = "z"

# Remove the default panic handler
panic = "abort"
EOF

$ du -sh target/release/helloworld
372k    target/release/helloworld
```

```
$ cat > Cargo.toml <<EOF
[profile.release]
# Strip your binaries of debug information
strip = true

# Optimize for size
opt-level = "z"

# Remove the default panic handler
panic = "abort"
EOF

$ du -sh target/release/helloworld
372k    target/release/helloworld
```

```
$ cat > Cargo.toml <<EOF
[profile.release]
# Strip your binaries of debug information
strip = true

# Optimize for size
opt-level = "z"

# Remove the default panic handler
panic = "abort"
EOF

$ du -sh target/release/helloworld
372k    target/release/helloworld
```

```
$ cargo install cargo-bloat
$ cargo bloat --release --crates
    Finished release [optimized] target(s) in 0.00s
    Analyzing target/release/helloworld

File  .text     Size Crate
5.8%  98.1% 258.7KiB std
0.0%   0.0%      83B [Unknown]
0.0%   0.0%      56B helloworld
5.9% 100.0% 263.6KiB .text section size, the file size is 4.4MiB
```

Get rid of `std`

```
#![no_std]
#![no_main]

use libc_print::libc_println;

#[no_mangle]
pub extern "C" fn main() -> isize {
    libc_println!("Hello World");
    0
}

#[cfg(not(test))]
use core::panic::PanicInfo;

#[cfg(not(test))]
```

Get rid of `std`

```
 6 #[no_mangle]
 7 pub extern "C" fn main() -> isize {
 8     libc_println!("Hello World");
 9     0
10 }
11
12 #[cfg(not(test))]
13 use core::panic::PanicInfo;
14
15 #[cfg(not(test))]
16 #[panic_handler]
17 fn panic(_panic: &PanicInfo<'_>) -> ! {
18     loop {}
19 }
```

```
$ cargo bloat --release --crates
   Compiling helloworld v0.1.0 (/home/thomas/embdlinux/helloworld)
    Finished release [optimized] target(s) in 0.24s
    Analyzing target/release/helloworld

File  .text   Size Crate
0.1%  65.6%   836B std
0.0%  10.0%   128B [Unknown]
0.0%   7.8%   100B libc_print
0.2% 100.0% 1.2KiB .text section size, the file size is 813.5KiB

$ strip target/release/helloworld
    && du -sh target/release/helloworld
16K     target/release/helloworld

$ objdump -h -j .text  target/release/helloworld
target/release/helloworld:     file format elf64-x86-64

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
 13 .text         000004fb  0000000000001040  0000000000001040  00001040  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
```

1275 bytes of code

(0x4fb)

The same program in C

```
$ cat > main.c
#include <stdio.h>

int main() { printf("Hello World\n"); }

$ gcc -o main -Os main.c
$ strip main

$ ls -al main target/release/helloworld
-rwxrwxr-x 1 thomas thomas 14472 Mar 26 23:29 main
-rwxrwxr-x 2 thomas thomas 14400 Mar 26 23:18 target/release/helloworld
```

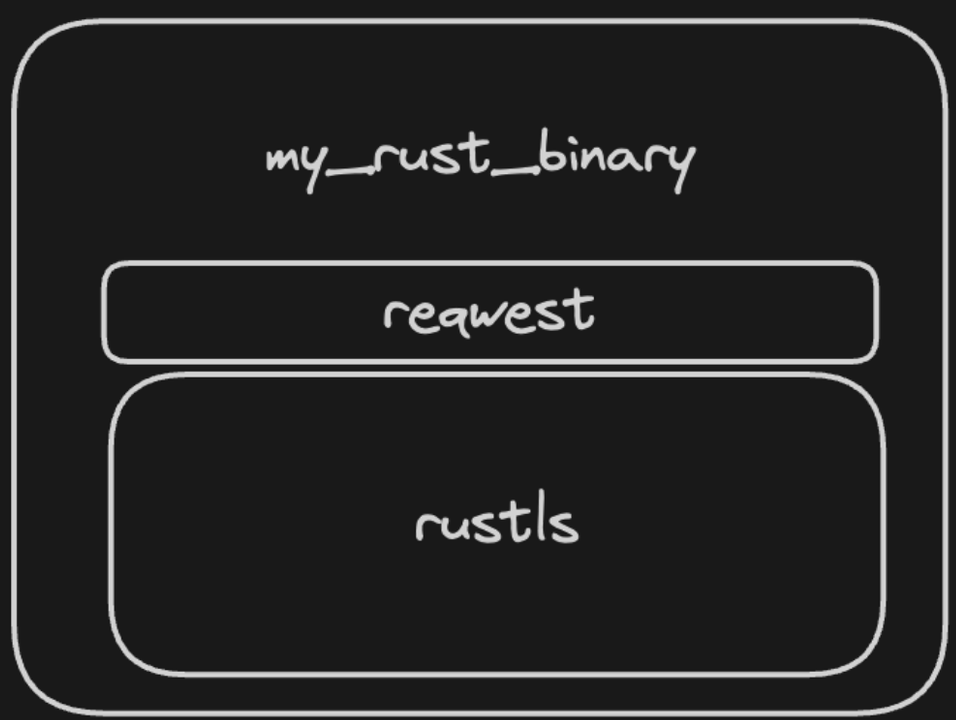
Rust binary is smaller than C!!!

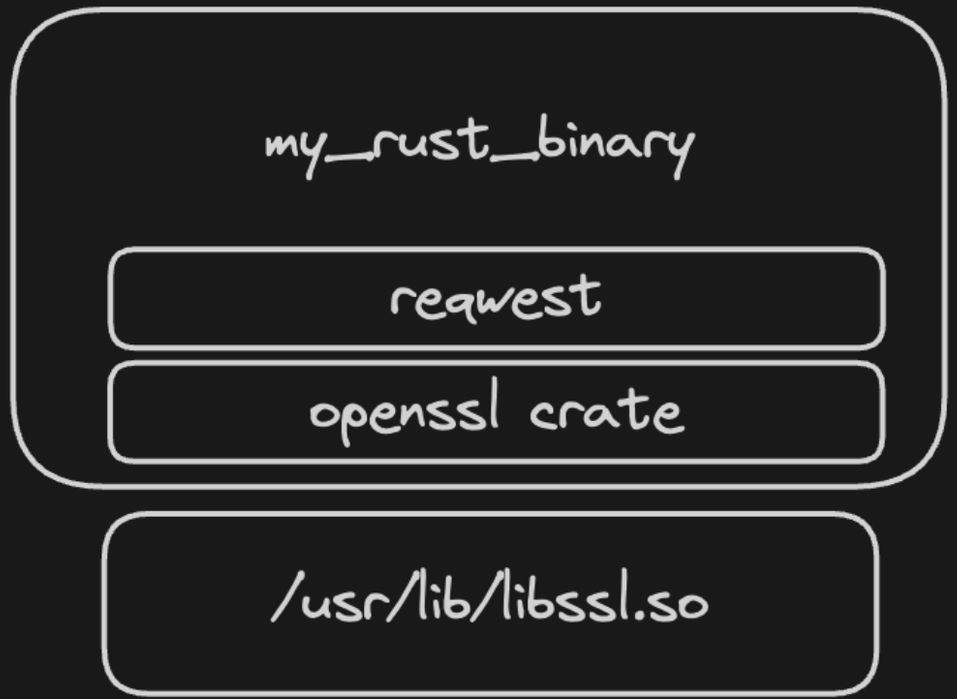
Take aways

- Rust’s std is about 260 kB
- It’s statically linked in each rust binary
- You can …
  - Do without std
  - Compile it from source – Enables better dead-code removal and optimizing for size
  - Use a busybox approach to avoid multiplying copies of rust::std on disk

```
optimize() // more
```

Adding libraries to your project...

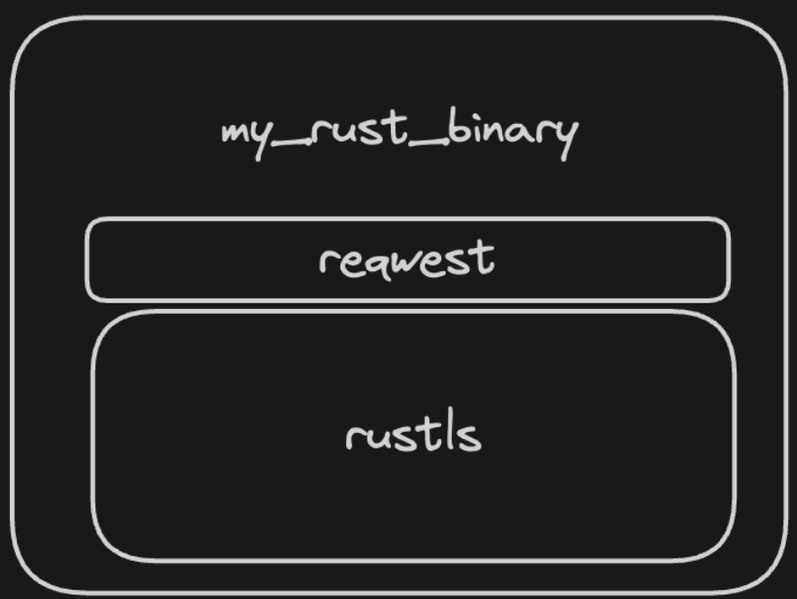




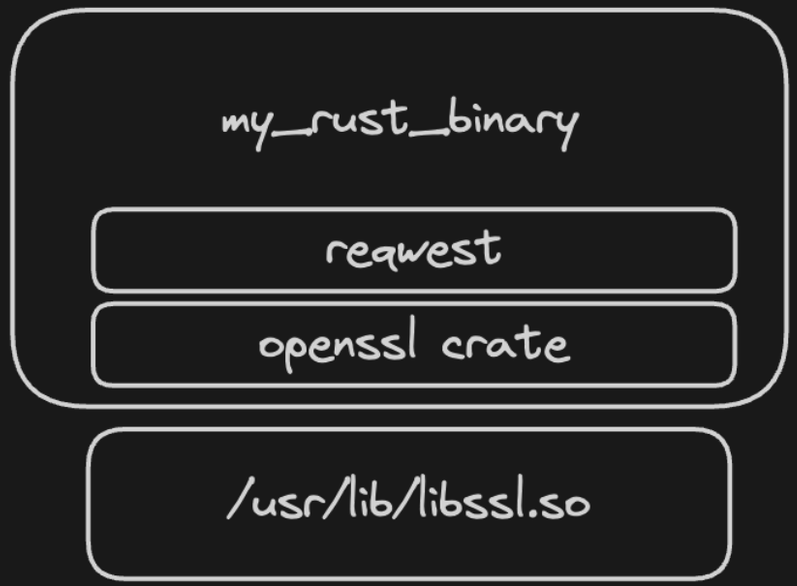
```rust
// main.rs
fn main() {
    let body = reqwest::blocking::get("https://www.rust-lang.org")
        .unwrap()
        .text()
        .unwrap();

    println!("body = {body:?}");
}
```

```
[features]
rust-tls = ["reqwest/rustls-tls"]
openssl-tls = ["reqwest/native-tls"]
```

```
$ cargo build --release --features=rust-tls
$ cargo build --release --features=openssl-tls
```



2.4 MB



1.5 MB

667 kB

2.2 MB

 18 months in

We love 🦀

Our code reviews are so much more interesting

You will never read&write C code the same way

EMBEDDED
LINUX